Advanced Regression 4b: Machine learning, decision trees

Garyfallos Konstantinoudis

Mar 14, 2025

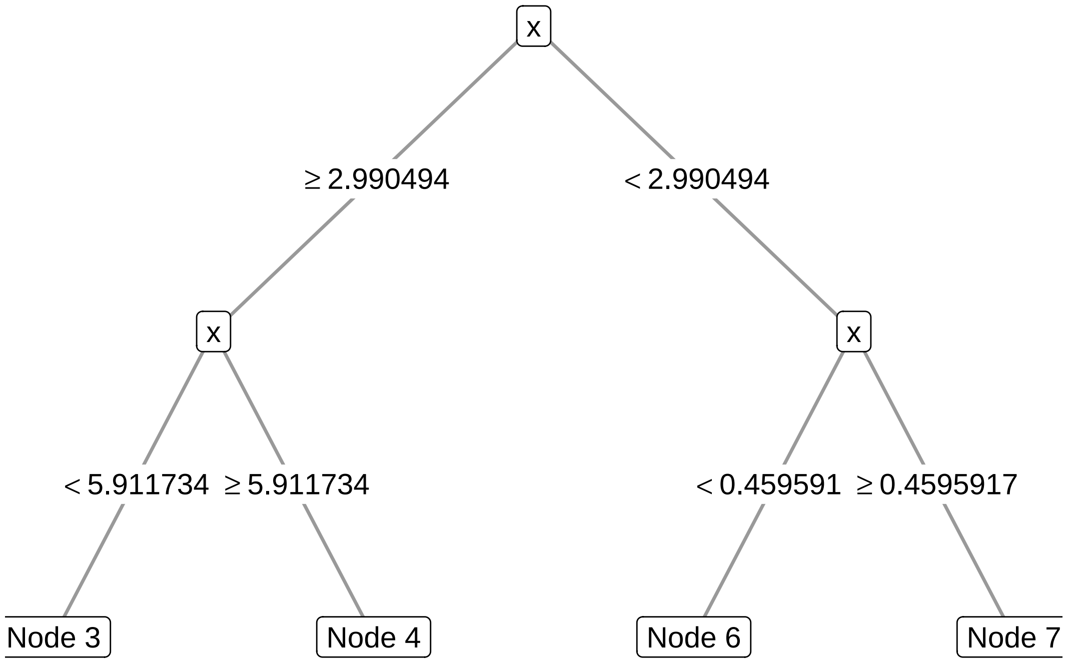
- Motivation for decision trees
- Technical definition
- Decision trees in `R`

Decision trees and ensemble methods

- **Decision tree**: A single tree
- **Bagging**: A meta-algorithm over trees
- **Random forest**: A meta-algorithm over random trees
- **Boosting**: A meta-algorithm over sequential trees

Decision trees: an introduction

```
Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
i Please use `linewidth` instead.
```



Decision trees: an introduction

- Decision trees are drawn upside down.

Decision trees: an introduction

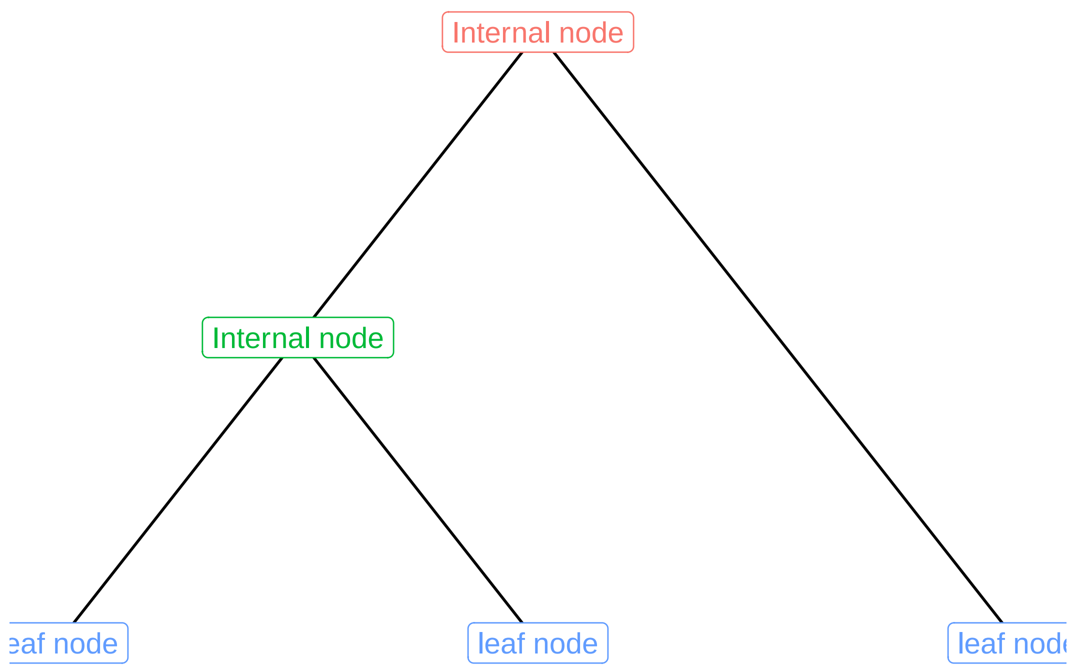
Notation:

- **Nodes** or **splits**: Points along the tree where the predictor space is split.
- **Leaves**: Terminal nodes
- **Branch**: Segments of a tree that connect the nodes

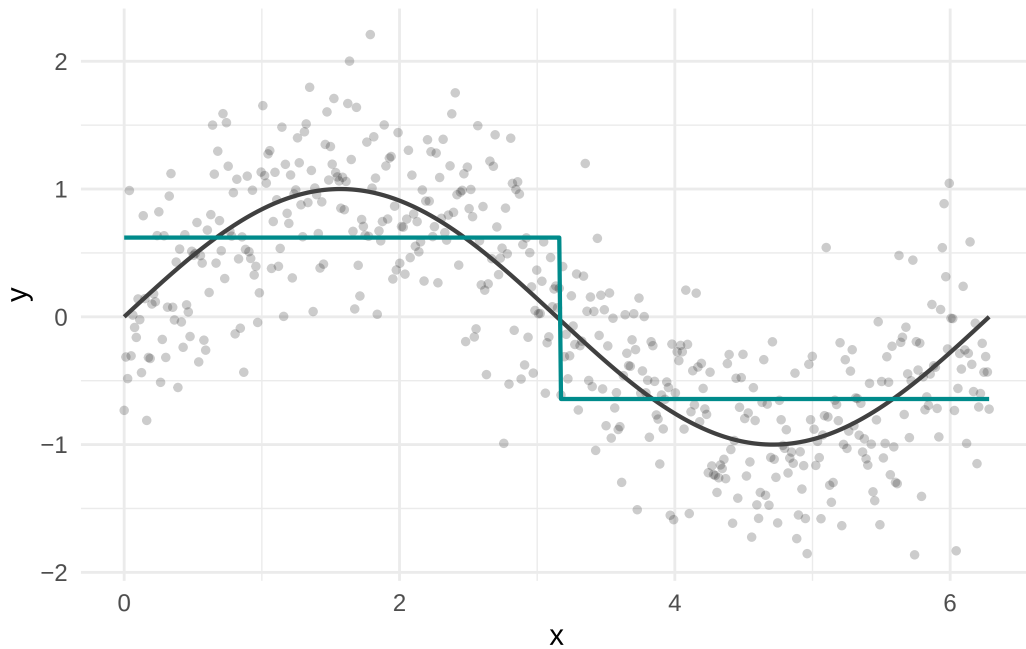
Outcomes:

- **Quantitative**: Regression trees
- **Categorical**: Classification trees considering $k = K$ categories

Decision trees: an introduction



Decision trees: another example



Problem: How to select the partition?

How to fit a decision tree?

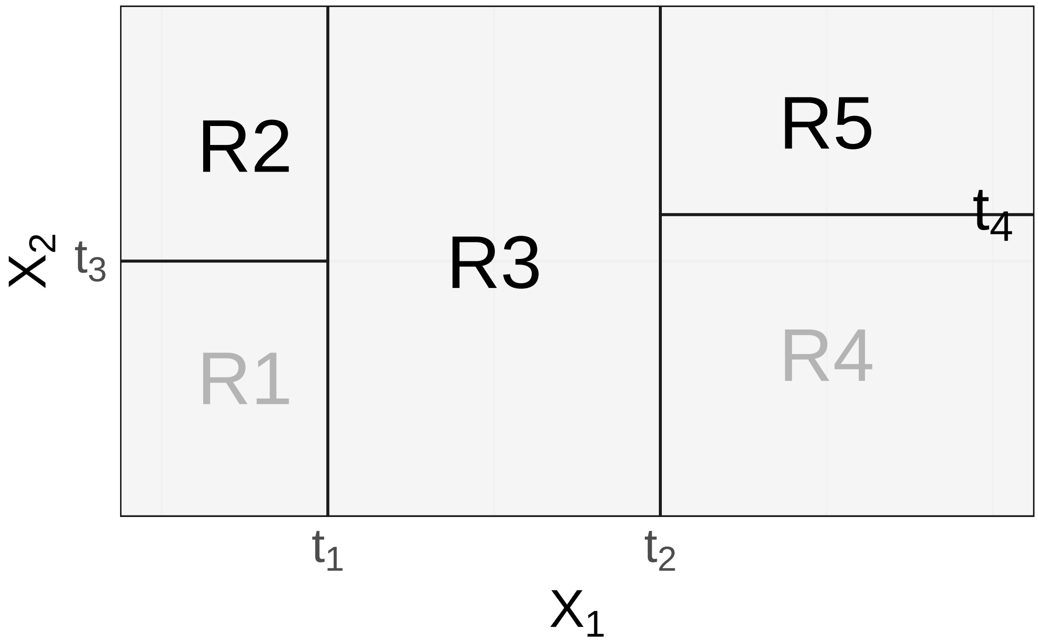
1. Divide the predictor space $(x_1, x_2, ..., x_p)$ into J distinct and non-overlapping regions, $r_1, r_m, ..., r_M$, where $m \in 1, ..., M$.
2. For every observation that falls in the same region r_m we make the same prediction based on the mean (median) of all observations in region r_m.
3. Define regions $r_1, r_2, ..., r_M$ to minimise the residual sum of squares

$$RSS = \sum_{m=1}^{M} \sum_{i \in m} (y_i - \bar{y}_m)^2$$

- Algorithm: Recursive binary splitting

Exercise: Reconstruct the tree

```
Warning in is.na(x): is.na() applied to non-(list or vector) of type
'expression'
```

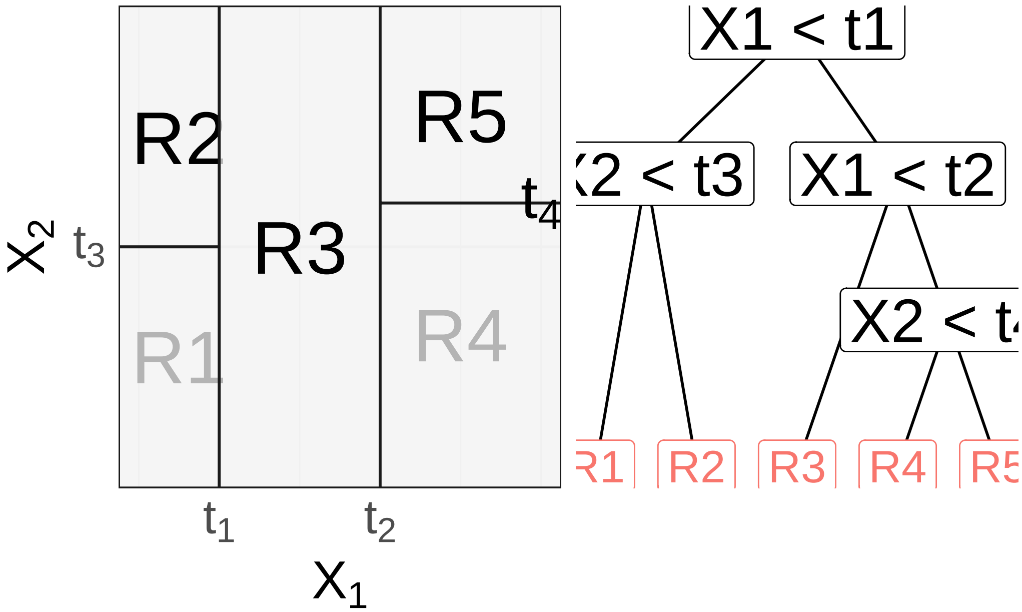


- Assume we have two variables, X_1 on the x-axis and X_2 on the y-axis.
- R_1 to R_5 map out a partition.
- t_1 to t_4 are the split values.

Reconstruct the respective tree

Exercise: Reconstruct the tree

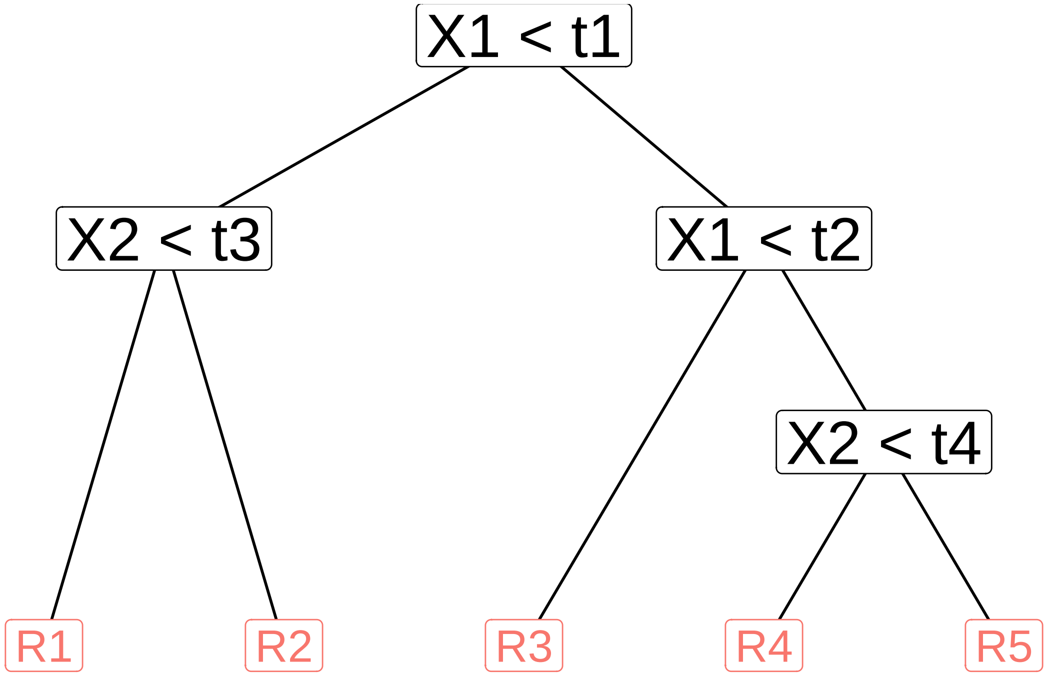
```
Warning in is.na(x): is.na() applied to non-(list or vector) of type
'expression'
```



Implementation

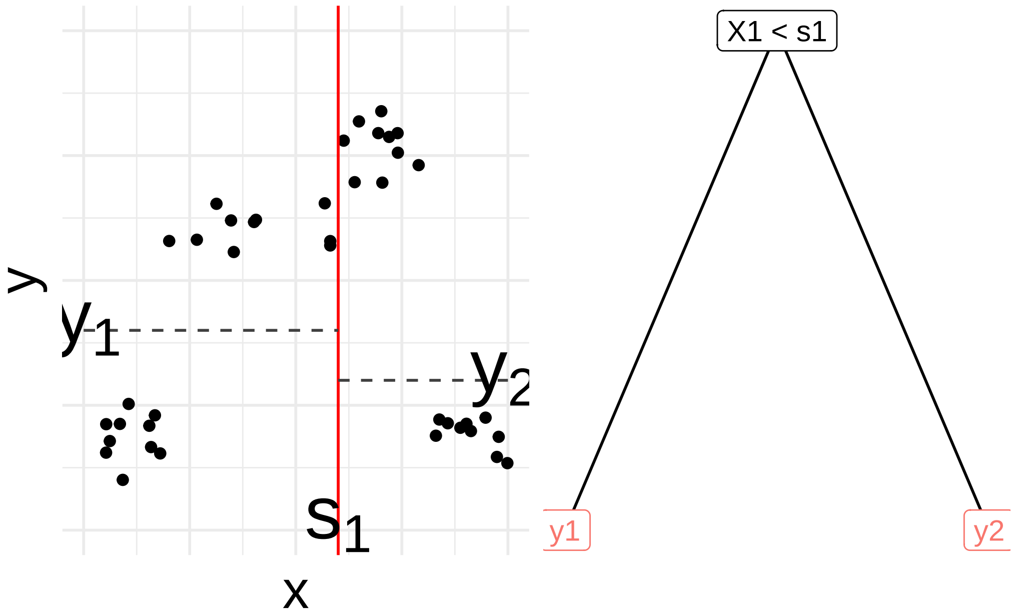
For each variable x_k:

- Find the optimal cutoff point t :
  -
$$\min_s \text{MSE}(y_i|x_{ik} < t) + \text{MSE}(y_i|x_{ik} \geq t)$$
- Choose variable yielding lowest MSE
- Stop when MSE gain is too small



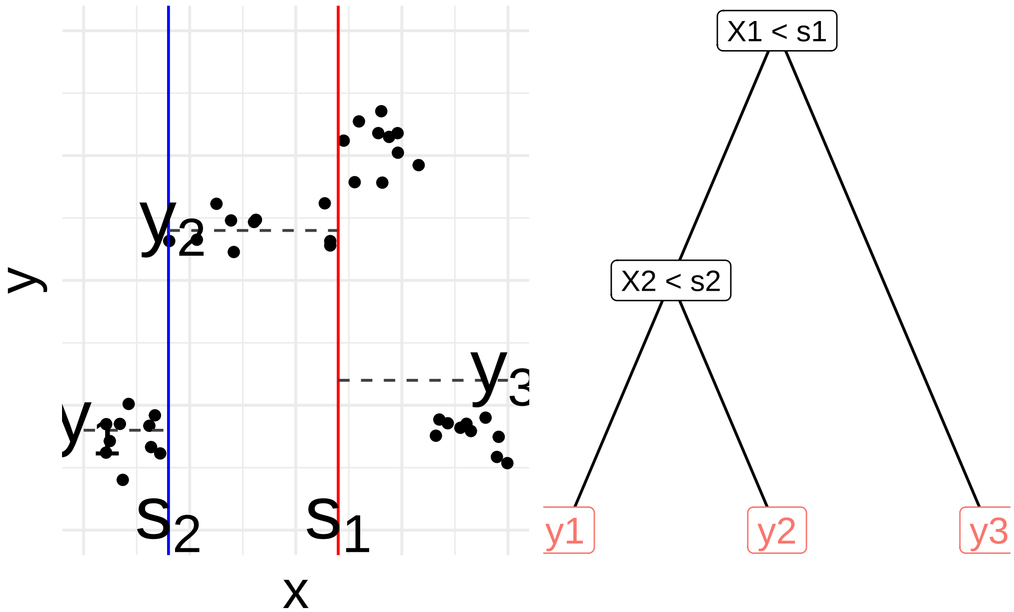
Example 1

```
Warning in is.na(x): is.na() applied to non-(list or vector) of type
'expression'
Warning in is.na(x): is.na() applied to non-(list or vector) of type
'expression'
Warning in is.na(x): is.na() applied to non-(list or vector) of type
'expression'
```



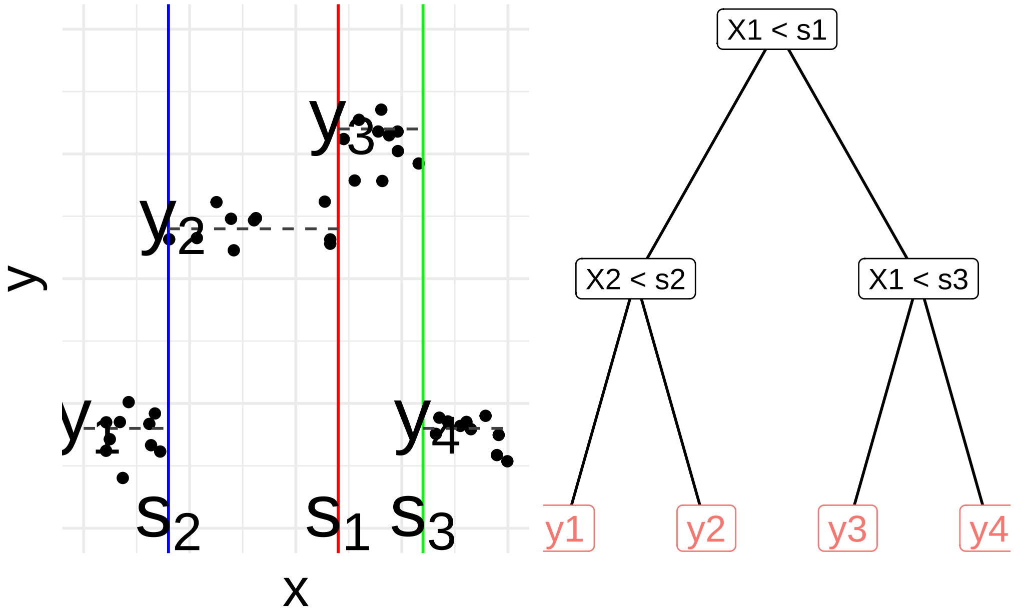
Example 1

```
Warning in is.na(x): is.na() applied to non-(list or vector) of type
'expression'
Warning in is.na(x): is.na() applied to non-(list or vector) of type
'expression'
Warning in is.na(x): is.na() applied to non-(list or vector) of type
'expression'
Warning in is.na(x): is.na() applied to non-(list or vector) of type
'expression'
Warning in is.na(x): is.na() applied to non-(list or vector) of type
'expression'
```



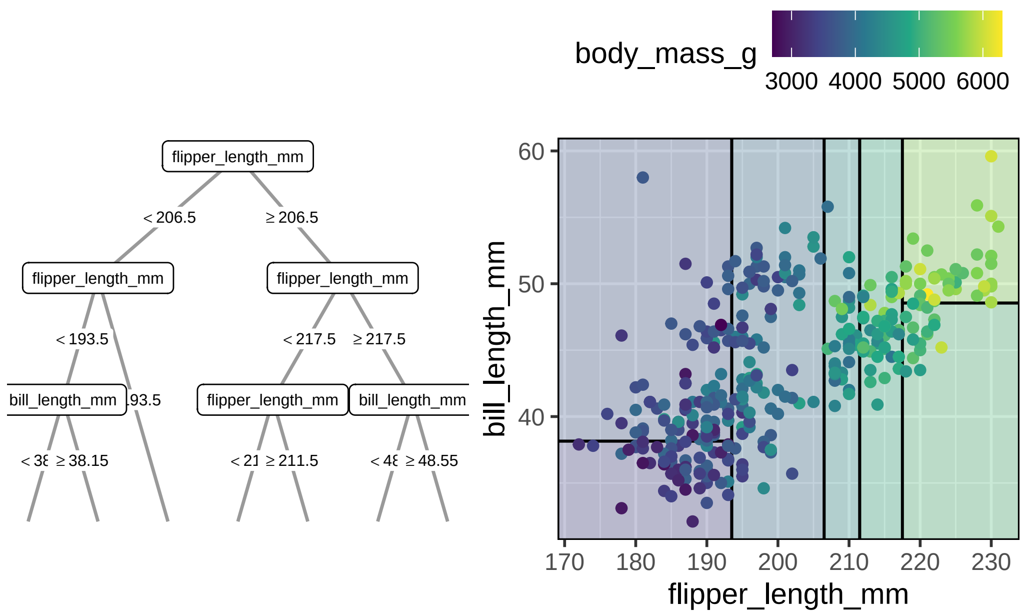
Example 1

```
Warning in is.na(x): is.na() applied to non-(list or vector) of type
'expression'
Warning in is.na(x): is.na() applied to non-(list or vector) of type
'expression'
Warning in is.na(x): is.na() applied to non-(list or vector) of type
'expression'
Warning in is.na(x): is.na() applied to non-(list or vector) of type
'expression'
Warning in is.na(x): is.na() applied to non-(list or vector) of type
'expression'
Warning in is.na(x): is.na() applied to non-(list or vector) of type
'expression'
Warning in is.na(x): is.na() applied to non-(list or vector) of type
'expression'
```



Example 2

```
Warning: Removed 2 rows containing missing values or values outside the scale range
(`geom_point()`).
```



Measures for model fit (node impurity)

Classification trees:

- **Gini index** of leaf m:

$$G_m = \sum_{k=1}^{K} p_{mk}(1 - p_{mk}),$$

  p_{mk}: proportion of observations in region R_m of class k.

- **Entropy** of leaf m

$$D_m = -\sum_{k=1}^{K} p_{mk} \log(p_{mk})$$

Measures for model fit (node impurity)

Regression trees, use **deviance** in leaf m

$$\text{dev}_m = \sum_{i \in m} (y_i - \mu_m)^2$$

where

- $i \in m$: Individuals in leaf m
- μ_m: Mean in leaf m

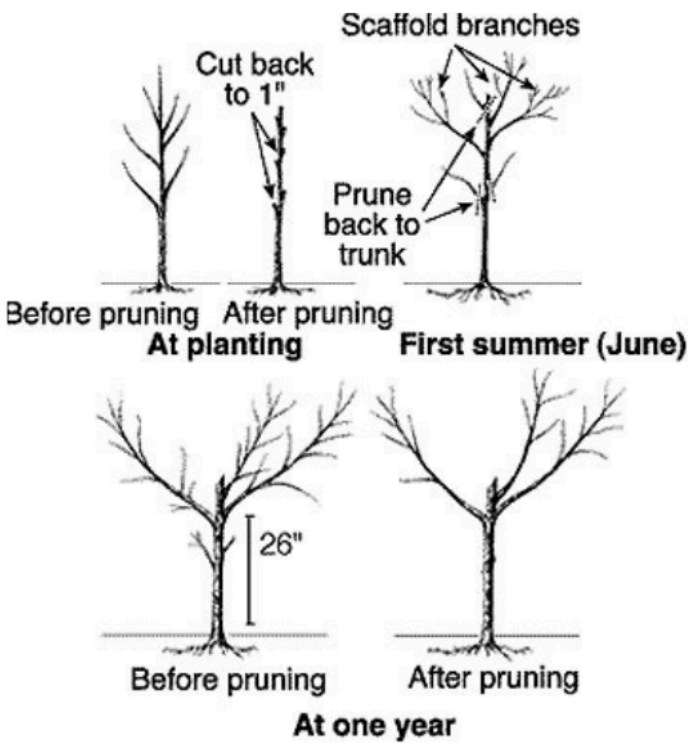
Overfitting

- Regression trees tend to overfit
- In principle, they could assign each observation to one leaf

Tree pruning

A smaller tree with fewer splits may generalise better to new observations.

Solution: **Pruning**



Tree pruning

A smaller tree with fewer splits may generalise better to new observations.

Cost complexity pruning or weakest link pruning: Find tree T

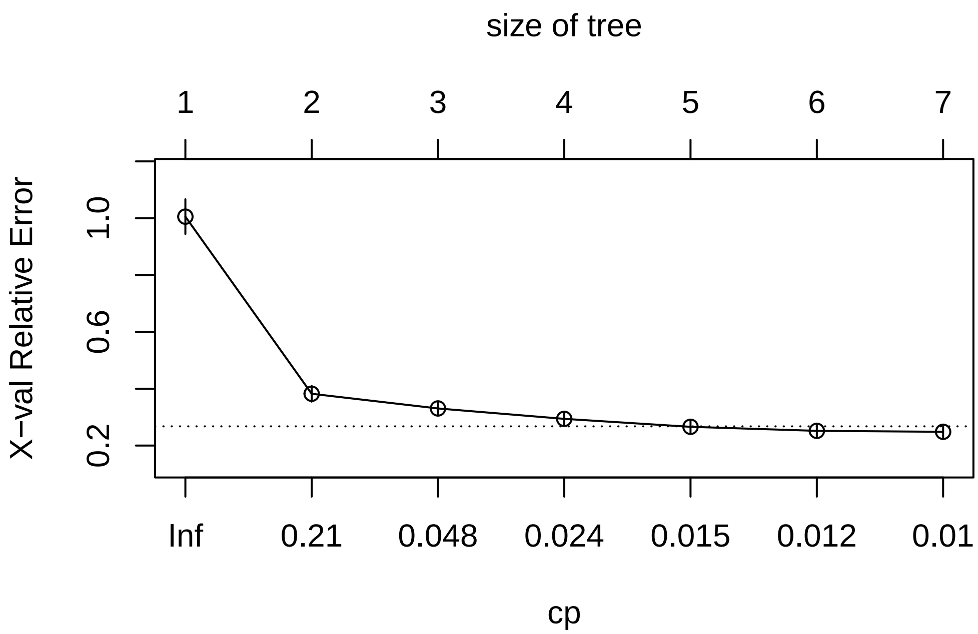
$$\underset{T}{argmin} \sum_{m=1}^{|T|} \sum_{i \in m} (y_i - \mu_m)^2 + \alpha \mid T \mid$$

where $\mid T \mid$ is the number of leaves and α a regularisation parameter.

Tree pruning

Select the regularisation parameter α (cp in plot below) that produces the tree with the lowest node impurity (measured by deviance) as evaluated by cross-validation.

```
plotcp(tree2)
```



Decision trees in R: rpart

```
fit <- rpart(y ~ x1 + x2, data = df, control = list(
  cp = 0.01, # Any split that does not decrease the overall lack of fit by a factor of cp is
  minbucket = 5, # minimum number of observations in a terminal node
  maxdepth = 4, # maximum depth of any node
  xval = 10 # number of cross validation splits
))

summary(fit)
```

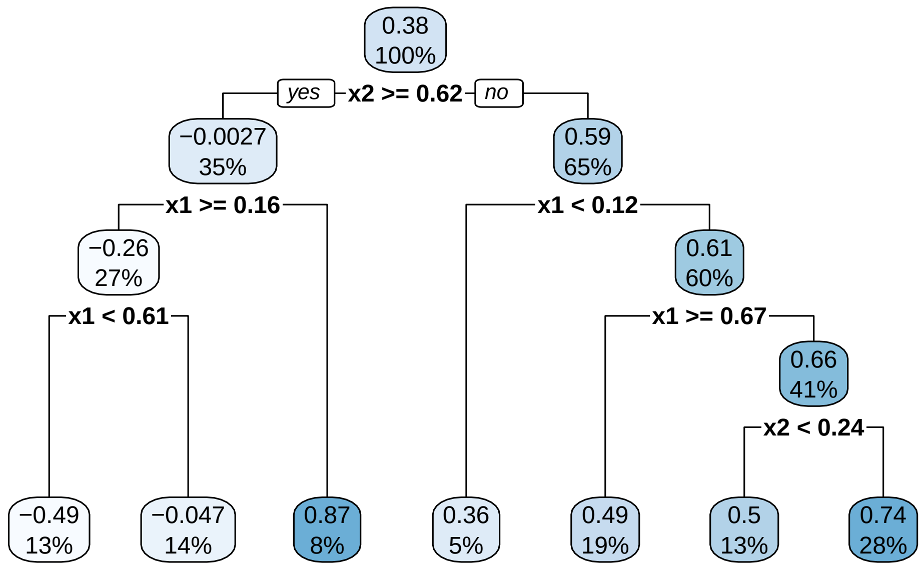
```
Call:
rpart(formula = y ~ x1 + x2, data = df, control = list(cp = 0.01,
```

```
    minbucket = 5, maxdepth = 4, xval = 10))
  n= 100

          CP nsplit rel error    xerror      xstd
1 0.32003170      0 1.0000000 1.0236472 0.1308996
2 0.31470048      1 0.6799683 1.0401587 0.1506900
3 0.05304048      2 0.3652678 0.5376809 0.1263461
4 0.01621269      3 0.3122273 0.5114605 0.1241012
5 0.01000000      6 0.2635893 0.5912215 0.1314069

Variable importance
x1 x2
55 45

Node number 1: 100 observations,    complexity param=0.3200317
  mean=0.382746, MSE=0.2500067
  left son=2 (35 obs) right son=3 (65 obs)
  Primary splits:
      x2 < 0.6196249  to the right, improve=0.32003170, (0 missing)
      x1 < 0.1637487  to the right, improve=0.05364018, (0 missing)
  Surrogate splits:
      x1 < 0.09421782 to the left,  agree=0.69, adj=0.114, (0 split)

Node number 2: 35 observations,    complexity param=0.3147005
  mean=-0.002727879, MSE=0.3076941
  left son=4 (27 obs) right son=5 (8 obs)
  Primary splits:
      x1 < 0.1637487  to the right, improve=0.7305700, (0 missing)
      x2 < 0.8966343  to the right, improve=0.1226793, (0 missing)

Node number 3: 65 observations,    complexity param=0.01621269
  mean=0.5903089, MSE=0.09585185
  left son=6 (5 obs) right son=7 (60 obs)
  Primary splits:
      x1 < 0.1247155  to the left,  improve=0.04677003, (0 missing)
      x2 < 0.4999134  to the right, improve=0.03307330, (0 missing)

Node number 4: 27 observations,    complexity param=0.05304048
  mean=-0.2608075, MSE=0.0961228
  left son=8 (13 obs) right son=9 (14 obs)
  Primary splits:
      x1 < 0.6146958  to the left,  improve=0.5109389, (0 missing)
      x2 < 0.763308   to the right, improve=0.2277072, (0 missing)
```

```
  Surrogate splits:
      x2 < 0.6489393  to the right, agree=0.593, adj=0.154, (0 split)

Node number 5: 8 observations
  mean=0.8682907, MSE=0.03828188

Node number 6: 5 observations
  mean=0.3583694, MSE=0.0489721

Node number 7: 60 observations,    complexity param=0.01621269
  mean=0.6096372, MSE=0.09490192
  left son=14 (19 obs) right son=15 (41 obs)
  Primary splits:
      x1 < 0.6728131  to the right, improve=0.06754766, (0 missing)
      x2 < 0.5724012  to the right, improve=0.05209399, (0 missing)
  Surrogate splits:
      x2 < 0.604362   to the right, agree=0.7, adj=0.053, (0 split)

Node number 8: 13 observations
  mean=-0.4907874, MSE=0.05958377

Node number 9: 14 observations
  mean=-0.04725461, MSE=0.0353342

Node number 14: 19 observations
  mean=0.4920235, MSE=0.05741783

Node number 15: 41 observations,    complexity param=0.01621269
  mean=0.6641411, MSE=0.1028915
  left son=30 (13 obs) right son=31 (28 obs)
  Primary splits:
      x2 < 0.2424118  to the left,  improve=0.12799780, (0 missing)
      x1 < 0.2395609  to the left,  improve=0.03963169, (0 missing)

Node number 30: 13 observations
  mean=0.4957193, MSE=0.06042682

Node number 31: 28 observations
  mean=0.7423369, MSE=0.1033228
```

Decision trees in R: rpart

```r
rpart.plot(fit)
```



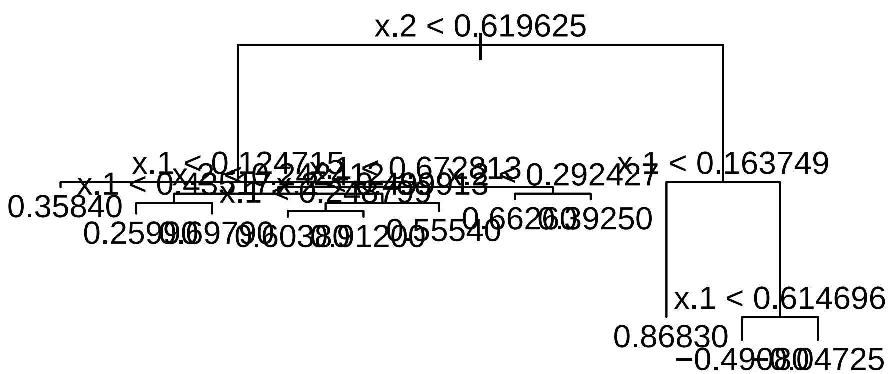
Decision trees in R: rpart

```r
# cp is complexity parameter to which the rpart object will be trimmed.
# trained with cp 0.01
prune(fit, cp = 0.1)
```

```
n= 100

node), split, n, deviance, yval
      * denotes terminal node

1) root 100 25.0006700  0.382746000
  2) x2>=0.6196249 35 10.7692900 -0.002727879
    4) x1>=0.1637487 27  2.5953160 -0.260807500 *
    5) x1< 0.1637487 8  0.3062551  0.868290700 *
  3) x2< 0.6196249 65  6.2303700  0.590308900 *
```

Decision trees in R: tree

```r
x <- cbind(df$x1, df$x2) %>% as.matrix()
tree.out = tree(y ~ x)
plot(tree.out)
text(tree.out)
```




```r
# tree.control(nobs, mincut = 5, minsize = 10, mindev = 0.01)
```

Decision trees in R: tree

```r
cv.tree(tree.out)
```

```
$size
[1] 11 10  9  4  3  2  1

$dev
[1] 12.43920 12.28504 12.02079 12.23266 11.99556 25.46932 25.46932
```

```
$k
[1]      -Inf 0.3223692 0.3499831 0.4598793 1.3260477 7.8677233 8.0010073

$method
[1] "deviance"

attr(,"class")
[1] "prune"         "tree.sequence"
```

```
# prune.tree(tree.out, best)
# prune.tree(tree.out, k)
```

Overview: decision trees

Advantages:

- Interpretability
- Intuitive, mirror human decision making
- Allowing for non-linear effects

Disadvantages:

- Overfitting is an issue
- Highly unstable and variable, small changes in the input data can cause big changes in the tree structure
- Minimal bias, but high variance

  Ensemble methods: Fit not one, but multiple trees.